# Functionality Screenshots

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

# README

**Functionality of Application:**

* The purpose of this application is to allow Grazioso Salvare a rescue-animal training company in Austin Texas to be able to quickly identify dogs within a desired training criteria, The application allows the user to filter through 3 rescue filters, “water rescue”, “wilderness rescue”, “disaster rescue” as well as a “reset” button to unfiltered the list, the application will also show a visual representation of data in the form of a pie chart and provide a geolocation map of each animal the user selects in the list. I also included a search bar to allow users to search by breed.

**Tools and Usage:**

* **Jupyter Dash – Dash** is a web application framework for python, it is very easy to learn and provides interactive components like the ones listed below to allow me as a developer to implement callbacks and change the displayed data based on the user inputs.
* **Dash Leaflet –** Component of dash to allow for the creation of the geolocation graph that is interactive.
* **Plotly Expressed –** Used for creating the pie chart and other graphs if desired.
* **Pandas –** Used to manipulate (convert) data and view it.
* **MongoDB –** Model component for the storage of data. MongoDB was used for its high flexibility and scalability. The use of query language allows for a quick and easy way to retrieve and store data. MongoDB is also stored in Binary JSON format allowing easy integration within python.
* **Dash Data Table –** A component of dash to allow for the creation of interactive data tables.

*Documentation Links*

* [MongoDB Documentation](https://www.mongodb.com/docs/)
* [pandas documentation — pandas 2.1.3 documentation (pydata.org)](https://pandas.pydata.org/docs/index.html)
* [Dash Documentation & User Guide | Plotly](https://dash.plotly.com/)
* [Dash Leaflet (dash-leaflet.com)](https://www.dash-leaflet.com/)
* [Dash DataTable | Dash for Python Documentation | Plotly](https://dash.plotly.com/datatable)

**Project Structure:**

**Dashboard.ipynb**

* + ***Class Connection*** – Import Animal Shelter Class from CrudFunction

Code example

import sys  
sys.path.append('/home/justinswinney\_snhu/Desktop/Python')  
from CrudFunction import AnimalShelter

* + ***Database Connection*** – Connecting to MongoDB by passing a username and password into the Animal Shelter class from the CrudFunction.py

Code example

username = "usernameHere"  
password = "passwordHere"  
db = AnimalShelter(username, password)

* + ***Data Retrieval and Conversion*** – Retrieving data from mongo DB through Animal shelter ReadMany method and converting it to Pandas data frame.

Code example

df = pd.DataFrame.from\_records(shelter.readMany({}))

* **Dashboard Layout** – Combination of HTML and Dash components (data table, pie chart, map) to create user interface.

Code example

html.Div(  
 style={'display': 'flex', 'justifyContent': 'space-between','alignItems':'flex-end'},  
 children=[  
 dcc.RadioItems(  
 id='filter-type',  
 options=[  
 {'label': 'Water Rescue', 'value': 'water\_rescue'},  
 {'label': 'Wilderness Rescue', 'value': 'wilderness\_rescue'},  
 {'label': 'Disaster Tracking', 'value': 'disaster\_tracking'},  
 {'label': 'Reset', 'value': 'reset'}  
 ],  
 inline=True,  
 labelStyle={'margin-right': '20px'}),  
 dcc.Input(  
 id='search-bar',  
 type='text',  
 placeholder='search breed..',  
 style={'margin-right': '10px'})  
 ]),

],  
 data=df.to\_dict('records'),  
 #*FIXME: Set up the features for your interactive data table to make it user-friendly for your client* page\_size = 10,  
 style\_table = {'overflowX': 'auto'},  
 row\_selectable = 'single',  
 selected\_rows = [0],  
 ),  
 html.Br(),  
 html.Hr(),  
 html.Div(  
 id='map-id',  
 className='col s12 m6',  
 )  
])

* Callbacks – Updating components data based on user inputs.

Code example

@app.callback(Output('datatable-id','data'),  
 [Input('filter-type', 'value'),  
 Input('search-bar', 'value')])

* Filters – Users select Radio buttons or search bar to decide what rescue animal they would like to filter through, conditional statement will check to see what user input (filter-type) is selected and make the appropriate queries to pull and update data table and graphs and maps or the display to the user.

def update\_dashboard(filter\_type, search\_query):  
 try:  
 df = pd.DataFrame.from\_records(db.readMany({}))  
 update\_graphs(df)  
 if filter\_type == 'reset':  
 df = pd.DataFrame.from\_records(db.readMany({}))  
 update\_graphs(df)# updating graph  
 elif filter\_type == 'water\_rescue':  
 water\_breeds = ["Labrador Retriever Mix", "Chesapeake Bay Retriever", "Newfoundland"]  
 query = createQuery("Dog", water\_breeds, "Intact Female", 26, 156)  
 df = pd.DataFrame.from\_records(db.readMany(query)) #Read all from Water Rescue  
 update\_graphs(df)# updating graph

## CREATE QUERY METHOD ###  
def createQuery(animal\_type, breeds, sex, min\_age, max\_age): # takes in type,breeds, sex,min and max age.  
 query = {"animal\_type": animal\_type,"breed":{"$in": breeds}, "sex\_upon\_outcome": sex, "age\_upon\_outcome\_in\_weeks": {"$gte": min\_age, "$lte": max\_age}}  
 return query

* Graph and Maps – Functions to update graphs and maps based on current data table filters.

Code example

def update\_graphs(viewData):  
 if viewData is None:  
 return []  
 return [  
 dcc.Graph(   
 figure = px.pie(viewData, names='breed', title='Preferred Animals')  
 )   
 ]

def update\_map(viewData, index):   
 if viewData is None:  
 return  
 elif index is None:  
 return  
   
 dff = pd.DataFrame.from\_dict(viewData)

**CrudFunction.py**

***Class Animal Shelter***

* + ***Database Connection*** - Establishes connection to MongoDB “AAC” collection animals, with hard coded username and password.

Code example

def \_\_init\_\_(self):  
 USER = 'aacuser'  
 PASS = 'SNHU1234'  
 HOST = 'nv-desktop-services.apporto.com'  
 PORT = 32304  
 DB = 'AAC'  
 COL = 'animals'  
 self.client = MongoClient('mongodb://%s:%s@%s:%d' % (USER,PASS,HOST,PORT))  
 self.database = self.client['%s' % (DB)]  
 self.collection = self.database['%s' % (COL)]  
 self.modified\_count = 0

* + ***ReadMany*** – Read method for pulling all documents within a collection that matches query.

Code example

def readMany(self, query):  
 cursor = self.collection.find(query) #creating cursor and querying dataase collection. using find and query variable to filter results.  
 listResult = list(cursor) #Converting results to a list.  
 return listResult

**Unused methods of CrudFunction within the Dashboard.**

* + *ReadOne* - Read method for pulling a single document within a collection that matches query.
  + *Create* – Create method for creating single document.
  + *Update* – Update method for updating all documents within a collection that match query.
  + *Delete* – Delete method for deleting all documents within a collection that match query.

**Challenges:**

* Python language structure, as someone who is primarily focused on Java, I found myself using semicolons and basic java syntax during the creation of this program causing errors, this challenge is not necessarily tackled, but the longer I spent coding the more I adapted.
* Code structures, I spent time reading documentation when trying to achieve desired results, for example styling radio buttons within the Dash library.
* Data review, I did not initially review the data provided in the excel sheet, once I created my first filter method and started testing, I noticed only 1 type of breed being displayed when I was searching for 3 types. I automatically assumed my code had an error and began trouble shooting before finally reviewing the data and seeing that only one type was going to populate as the others were not pure breeds that met the read query, this taught me the true value of reading all documents before diving into a project.